INFORME TÉCNICO SMARTSAT 2.0

Autores

Yury Andrea Castiblanco Romero

Brian Steven Solano Castro

Francisco Javier Trujillo Sánchez

Universidad Piloto de Colombia
Escuela de ingenierías TIC
Semillero de investigación SmartTIC
Bogotá D.C., noviembre de 2018

INFORME TÉCNICO SMARTSAT 2.0

Yury Andrea Castiblanco Romero
Brian Steven Solano Castro
Francisco Javier Trujillo Sánchez

Asesor
Ing. Henry Artuuro Bastidas Mora M.Sc

Informe técnico para optar por el título de:
Ingeniero de Telecomunicaciones

Universidad Piloto de Colombia
Semillero de investigación SmartTIC
Bogotá D.C., noviembre de 2018

Tabla de Contenidos

Contenido

Abstract 1

Introducción 2

Justificación 3

1. Planteamiento del problema 4

2. Objetivos 5

2.1 Objetivo General 5

2.2 Objetivos específicos 5

3. Marco Teórico 6

4. Sistemas 9

4.1 Sistema de energía 9

4.2 Sistema de sensores 9

4.3 Sistema mecánico: 11

4.4 Sistema de comunicación: 12

5. costos smartsat 2.0 13

6. configuraciones de software y pruebas de funcionamiento 13

6.1 Arduino nigthly 13

6.1.1 Código DTH11 14

6. 1.2 Código BMP280 15

6.1.3 Código GYML 8511 16

6.1.4 GY GPS6MV2: 17

6.1.5 Servo SG90: 18

6.2 MÓDULOS XBEE 19

6.3 Fire base, node js y visual studio code 22

6.4 XCTU 28

6.5 Sistema de propulsión 30

6.6 Base de datos para mostrar las mediciones obtenidas 38

6.7 Pruebas de peso y transmisión: 44

7. Conclusiones 46

8. Limitaciones 46

9. Referencias bibliográficas 47

Aceptación del tutor

_____

Abstract

Over the years are reflected the climatic changes that take place on the planet earth, pollution increases, temperatures increase causing fires in some areas, carbon dioxide levels affect the health of the population with respiratory and cardiovascular problems.

Due to this problem around the world, it has become aware and that is why in 2016 Colombia participated, together with many countries, in the environmental agreement signed in Paris.

this is how the idea comes about. the smartSAT 2.0, which is a type of cansat satellite which function is measuring meteorological variables such as temperature, humidity, so contributing with reduction of environment pollution, it has a database fire protection, when the temperature sensor detects fire, it shows the exact point in a google map immediately, therefore we can proceed quickly to turn it off, performs wireless data transmission through xbee modules, it has a mini camera that captures video, and also(es para evitar que piensen que estan hablando de la camara) has its own propulsion system through a rocket that is shooted by a mixture of water and air.

Introducción

Un CANSAT es la simulación de un satélite real con sistema de sensores, de energía, de comunicación, y lanzamiento, cuyo tamaño es el de una lata de refresco de 375 mL, su peso no puede superar los 350 gr, y debe cumplir con una misión específica, el CANSAT debe hacer recolección de datos y volver a la superficie intacto para poder hacer el análisis de los datos recopilados, el término se utilizó por primera vez en Estados Unidos donde un grupo de estudiantes con ayuda del docente Twiggs quisieron elevar una carga pequeña funcional con estas especificaciones.

En este artículo se puede evidenciar todo el proceso realizado para desarrollar el satélite tipo Cansat "SmartSAT 2.0", el cual lleva a cabo una misión meteorológica, de observación de tierra, posicionamiento y detección de incendios, cuya programación está basada en c++ a través de la placa arduino Uno, a ella está ligada la programación de los sensores, los cuales a su vez envían los datos capturados a través de un módulo xbee configurado como router y son recibidos en tierra por un segundo módulo xbee configurado como coordinador, éste módulo captura los logs en tiempo real de los datos leídos por los sensores y luego son enviados a una base de datos en donde los muestra en forma de tabla.

Igualmente, se tiene una base de datos contra incendios a través de firebase, ésta recibe datos de posicionamiento y temperatura para saber el punto exacto en el que ocurre un incendio mostrándolo a través de un mapa de google.

Se construyó un sistema de propulsión que consiste en un cohete de agua, el cual es expulsado por medio de aire y agua que se bombea por un diseño realizado con tubos de agua caliente

Justificación

El presente proyecto se hace con el fin de contribuir de manera significativa al medio ambiente, debido a la composición de los elementos externos y misión del SmartSAT 2.0, la cual se encargará de la recolección de datos meteorológicos y visualización en tiempo real de los mismos, brindando así, el acceso de la información a aquellos usuarios que cuenten con la comunicación al servidor y a su vez, la información obtenida pueda ser manejada para uso libre y enfocada en los estudios cotidianos del comportamiento del ambiente en los diferentes espacios a evaluar y la variación de los estados; adicional, contando con la cámara equipada en el SmarSAT fortalecerá y brindará a la sociedad datos en pro de la prevención de incendios.

Alrededor del mundo anualmente se realizan varios concursos CANSAT con diferentes misiones, una de las más grandes es la competencia realizada por la ESA ((Agencia Espacial Europea) para personas entre los 14-20 años, asimismo países como Japón, Estados Unidos, y en América Latina Perú, Argentina, México se han unido a estos eventos, el objetivo de todos los concursos es poder diseñar un satélite tan pequeño como una lata de gaseosa y que pese máximo 350 gr.

Por supuesto Colombia no se queda atrás y en 2018 ha empezado a realizar este tipo de eventos, en abril del presente año la universidad Distrital Francisco José de Caldas realizó un reto de innovación IEEE “CANSAT para la paz” el cual consistía en realizar un prototipo de un satélite tipo CANSAT, el evento fue muy concurrido y desde niños hasta personas que no se encontraran vinculadas a una institución pero con deseos de participar pudieron inscribirse; Asimismo la universidad EAFIT de Medellín en alianza con la universidad Nacional, Universidad de Antioquia, la UPB y el ITM a la, participó en el reto Global Space Balloon Challenge, realizado por la universidad de Michigan (MIT), en este reto se pretendían alcanzar las máximas alturas posibles dentro de la estratósfera y contó con el apoyo de empresas como Postobón, RCN y TCC.

1. Planteamiento del problema

Con el paso de los años se ven reflejados los cambios climáticos que tienen lugar en el planeta tierra, aumenta la polución, aumentan las temperaturas ocasionando incendios en algunas zonas, los niveles de dióxido de carbono afectan la salud de la población con problemas respiratorios y cardiovasculares.

Debido a esta problemática alrededor del mundo se ha ido tomando conciencia y es por eso que en el año 2016 Colombia participó junto con muchos países en el acuerdo ambiental firmado en París, el cual busca que la temperatura media del planeta tierra no exceda los 2°C, en este tratado Colombia hizo compromisos como: "reducir un 20% las emisiones de gases de efecto invernadero, adaptar al país ante los cambios climáticos y dirigirse a una economía baja en carbono, entre otros." (Noticias cambio climático, Ministerio de Ambiente y Desarrollo sostenible, Disponible en: https//www.minambiente.gov.co)

Bogotá, Capital de Colombia cuenta con alrededor de 8.081.047 habitantes, y según un análisis realizado por el IDEAM en el año 2017, la capital lidera el ranquin de ciudades vulnerables al cambio climático, es por eso que surge la necesidad crear mecanismos que ayuden a monitorear las variables meteorológicas y poder controlarlas de una manera eficiente.

Es así como surge una nueva alternativa para monitoreo de variables meteorológicas como lo son los satélites tipo CANSAT ideales por su tamaño para recopilar este tipo de información y poder contribuir con el cuidado del medio ambiente, y debido a esto se plantea el siguiente interrogante ¿Cómo diseñar y construir un satélite tipo CANSAT para medición de temperatura, humedad, radiación UV, presión atmosférica, posicionamiento, captura de imágenes y que cuente con un sistema de lanzamiento propio?

2. Objetivos

2.1 Objetivo General

Diseñar y construir un satélite tipo CANSAT con el fin de llevar a cabo una misión meteorológica, de observación de tierra, posicionamiento y detección de incendios.

2.2 Objetivos específicos

1. Realizar un análisis de equipos y componentes para la correcta operación de cada uno de los subsistemas.

2. Generar el código de operación de los sensores sobre una placa de programación que permita la adquisición de datos a la computadora principal.

3. Proponer y construir un sistema de propulsión para el CANSAT.

4. Almacenar los datos obtenidos en la estación terrena en una base de datos local, para llevar a cabo la actualización de esta en un servidor virtual.

3. Marco Teórico

Estado del Arte

Un CANSAT es la simulación de un satélite real con sistema de sensores, de energía, de comunicación, y lanzamiento, cuyo tamaño es el de una lata de refresco de 375 mL, su peso no puede superar los 350 gr, y debe cumplir con una misión específica, es lanzado a una altura promedio entre los 50 y 100 metros, el CANSAT debe hacer recolección de datos y volver a la superficie intacto para poder hacer el análisis de los datos recopilados, el termino se utilizó por primera vez en Estados Unidos donde un grupo de estudiantes con ayuda del docente Twiggs quisieron elevar una carga pequeña funcional con estas especificaciones .

Alrededor del mundo se han hecho varias competencias de CANSAT, como por ejemplo en Europa la ESA (Agencia Espacial Europea) realiza este tipo de actividades para los miembros de diferentes estados, enfocado para estudiantes entre 14-20 años, conformados por grupos máximos de 6 estudiantes, para el año 2016 participaron en este evento países como Bélgica, Australia, Dinamarca, Luxemburgo, Estonia, Francia, Grecia, Italia, Reino Unido, entre otros.

Entre las reglas del concurso se estableció presentar un informe llamado CDR (Informe de Revisión de Diseño Crítico) en el cual se analizan las limitaciones del sistema y se garantiza que el CANSAT pueda cumplir con todos los requisitos que se prometieron. El lanzamiento de los CANSAT se realizó mediante cohetes que transportaban 2 CANSAT cada uno, y se lanzaron a 1 Km de altura, y por último se presentó un (CFR) informe final que contiene el resumen del proyecto.

En Argentina se ha realizado el proyecto CANSAT Argentina desde el 2004 gracias a la asociación de cohetería espacial experimental y modelista de Argentina, actualmente se realiza el proyecto en algunos colegios del país.

En algunos colegios se hacen incluso femto satélites, denominados así por pesar menos de 100 gr, se elevan a una altura de 100 mts y son controlados por un micro arduino. El 6 de diciembre de 2017 se realizó el décimo tercer lanzamiento en la agrupación Aeromodelista Pucará, en la cual el primer lanzamiento fue fallido debido a que no superó los 50 mts de altura, y un segundo lanzamiento tuvo éxito al superar los 150 mts de altura en el apogeo.

Perú no se queda atrás, pues Nohelia Merino Suasnábar, estudiante de ingeniería de la universidad UNI, con tan solo 22 años ganó una beca del Concytec para estudiar en el centro de investigaciones de la NASA al innovar con un CANSAT y desarrollar un Dron CANSAT el cual tiene una hélice que le permite volar al mismo tiempo que toma datos meteorológicos, capta fotografías panorámicas y diseña mapas de ecosistemas en 3D.

En México, la universidad UNAM también ha realizado una competencia de CANSAT desde el año 2008, para la primera versión se integró una misión de sentido social junto con beneficios para la población, los ganadores diseñaron un CANSAT basado en toma de imágenes infrarrojas de cultivos de un municipio para detectar daños en la vegetación, además de medir datos meteorológicos como temperatura, altitud, presión y posicionamiento GPS, las pruebas

se realizaron con un Drone que se elevó a una altura de 500 mts y el peso total del CANSAT ganador fue de 370 gr.

Asimismo, estudiantes de esta misma universidad han tenido participación en el concurso internacional CANSAT competition organizado por la NASA con un CanSat llamado Siqueiros, allí han competido con países como Estado Unidos, Canadá, Italia, Polonia y han ocupado el puesto 12 de la competencia. El objetivo central de esta competencia consistía en construir los componentes electrónicos para que el satélite pueda transmitir datos en tiempo real a la estación terrena, el CANSAT debería elevarse a alturas entre los 800-100 metros por medio de un cohete, funcionar con energía solar y debería recolectar los datos en 2 minutos aproximadamente. Para el año 2015 se llevó acabo el primero concurso nacional de CANSAT en México.

"En 2003 se formó el University Space Engineening Consortium (Unisec) en Japón, éste engloba a las universidades que realizan actividades aeroespaciales y espaciales del mundo. Lo que facilita la colaboración e intercambio entre las universidades a nivel mundial mediante cursos de entrenamiento CANSAT como el CANSAT Leader Training Program (CLTP) y capítulos universitarios (Sahara y Ando, 2013)." (Ciencia UNANL, Revista de divulgación científica y tecnológica de la universidad Autónoma de Nuevo León, edición 2085, Recuperado de http://cienciauanl.uanl.mx/?p=6442)

En Estados Unidos anualmente se realiza una competencia de CANSAT para estudiantes universitarios gracias a la American Astronautical Society (ASS) y el American Institute of Aeronautics and Astronautics (AIAA), además cuenta con el apoyo del Laboratorio Naval de Investigación (NRL), la Aeronáutica Nacional y la NASA. Generalmente se presentan alrededor de 20 equipos, pero cerca de 12 completan la misión, en los CANSAT se incluyen sensores de temperatura, humedad, deben aterrizar en un área designada y realizar teledetección.

En Turquía la competencia la organiza la Istanbul Technical University Cansat Team.

Colombia no es la excepción, el 12 y 13 de abril de 2018 se realizó el primer concurso en el país, llamado "Reto de innovación IEEE- CANSAT para la Paz 2018" en el cual se incluyeron estudiantes y aficionados, el objetivo del concurso fue realizar el diseño de un prototipo y exponerlo ante el jurado, algo significativo que cabe destacar en la competencia es que se incluyó una categoría llamada papagayos en donde se incluyeron alumnos de primaria, esto es muy importante para que desde ya en ellos se cree la cultura de participar en este tipo de eventos y conozcan de que se tratan estas tecnologías.

Este concurso fue organizado por la IEEE AESS, la Universidad Distrital Francisco José de Caldas, el Centro de Estudios y Ciencias Aeronáuticas (CEA)-Aerocivil, las ramas estudiantiles IEEE de la UD y la Universidad Javeriana, además de la Red de Investigación de Tecnología Avanzada- RITA UD, y se basó en la reglamentación de la ESA (Agencia Espacial Europea), en donde los grupos inscritos fueron de 3 a 5 alumnos o aficionados.

Contó con jurados como el señor Jorge Reynolds Pombo, creador del marcapasos, el señor Germán Puerta director del Planetario de Bogotá, el señor Raúl Andrés Joya docente de la Universidad Sergio Arboleda quien estuvo al frente del lanzamiento del único pico satélite lanzado al espacio en Colombia, entre otros.

“Se eligió el tema de la paz ya que en Colombia lo que tiene que ver con investigación aeroespacial en las condiciones meteorológicas ha estado sesgado por la confrontación y la guerra, sin embargo, ahora en un ambiente más pacífico la Universidad Distrital es líder en el proceso de paz y el posconflicto” (Reto de Innovación proyecto CanSat para la Paz Colombia 2018, UD, 05 de abril de 2018. Recuperado de: www.udistrital.edu.co/reto-innovacion-proyecto-cansat-colombia-para-paz)

Reto de Innovación proyecto CANSAT para la Paz Colombia 2018, UD, 05 de abril de 2018, www.udistrital.edu.co/reto-innovacion-proyecto-cansat-colombia-para-paz

Igualmente, la universidad EAFIT realizó alianza con estudiantes de la universidad Nacional, Universidad de Antioquia, la UPB y el ITM a la cual llamaron KRATOS, para participar en el reto Global Space Balloon Challenge, realizado por la universidad de Michigan (MIT), en este reto se pretenden alcanzar las máximas alturas posibles dentro de la estratósfera.

KRATOS contó con el apoyo de empresas como Postobón, RCN y TCC y denominó a su misión Simple-3, el CANSAT se lanzó el 27 de abril en la Dorada Caldas en la pista de la base Calderón de la Fuerza Aérea, el lanzamiento se realizó con un globo que se llenó con casi 13 metros cúbicos de hidrógeno, y dos pipetas y media de gas, el globo estalló a los 52.500 pies de altura, el Simple 3 se contaba con paneles solares, y cámaras de grabación de video y captura de imágenes.

“El mecanismo funciona de manera similar a las bisagras de una puerta y la mayoría del viaje los paneles van cerrados. Después de que el satélite pase la primera capa de la atmósfera, la troposfera, el equipo mandará una señal a través del sistema de telecomunicación en Tierra, para que sea procesada por el “cerebro” del CANSAT y mande una orden al sistema de apertura para que se despliegue”. (Proyecto Kratos, 5 de marzo de 2018. Recuperado de http://www.eafit.edu.co/escuelas/administracion/kratos/Paginas/globos.aspx)

La góndola que almacena el CANSAT está compuesta de fibra de carbono y fibra de vidrio, lo cual la hace muy resistente ante impactos, para la realización de esto el equipo KRATOS realizó un curso de laminación de fibra de vidrio con apoyo del SENA.

4. Sistemas

4.1 Sistema de energía

Baterías: El SmartSAT 2.0 cuenta con dos baterías de polímero y litio de 3.7V y 800mAh, las cuales están conectadas en serie para proporcionar 7.4 V.

Fig. 1. Batería polimero litio 3.7V

Regulador de voltaje LM2596: Este regulador tiene un voltaje de entrada de 4.5 a 40V y un voltaje de salida de 1.5 a 35V con una corrienta máxima de salida de 3ª, cuenta con un circuito regulado monolítico y el voltaje de salida es ajustable ideal para usarlo en microcontroladores como Arduino o Raspberry.

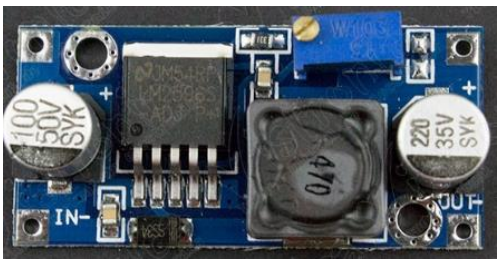


Fig. 2. Regulador de voltaje LM2596

4.2 Sistema de sensores

Para realizar las mediciones de las diferentes variables, se hizo uso del protocolo I2C y comunicación serial.

BMP280: Permite medir presión atmosférica, humedad y altitud.

Fig. 3. Imágen Sensor BMP 280

DTH11: es un sensor compuesto que contiene una señal digital calibrada de temperatura y humedad. El sensor incluye una sensación resistiva de componente húmedo y un dispositivo de medición de temperatura NTC, y está conectado con un microcontrolador de alto rendimiento de 8 bits.
Precisión de la medición de humedad: ± 5% RH
Precisión de medición de temperatura: ± 2 ℃

Fig. 4. Imágen Sensor DTH 11

GYML 8511: Sensor de luz Ultravioleta, salida analógica relacionada con la cantidad de luz detectada. Sensible a rayos UV-A y UV-B. Este índice se mide según la escala adoptada por la Agencia para la Protección del Medio Ambiente (EPA).

Fig. 5. Imágen Sensor GYML 8511

GY GPS6MV2: Tiene un módulo 6 U-Blox NEO de serie equipado en el PCB, una EEPROM, una pila tipo botón para mantener los datos de EEPROM intacta, un indicador LED, conectores y una antena de cerámica que permite transmitir los datos mediante comunicación serial.

Fig. 6. Imagen GPS GY GPS6MV2

El siguiente diagrama muestra la conexión de todos los elementos que usa el smartSAT 2.0:

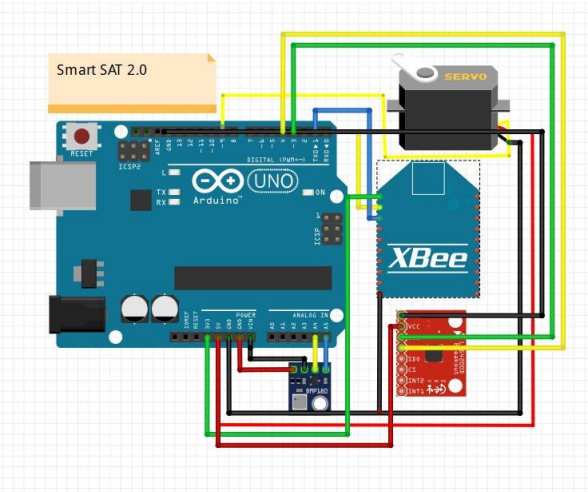


Fig. 7. Imágen conexión de SmartSAT 2.0

4.3 Sistema mecánico:

La estructura interna que da soporte a los elementos se se realizó mediante impresión 3D utilizando ácido poliláctico (PLA) de tipo flexible, un plástico cuya composición es a base de cereales (maíz y trigo), lo que lo hacen un material biodegradable y por lo tanto amigable con el medio ambiente. El PLA flexible proporciona una alta resistencia ante cualquier impacto. En la parte superior de la estructura se ubica el GPS y el módulo XBee para obtener mejor señal. En la parte inferior del SmartSAT 2.0 se ubican los demás sensores, el cbleado, el arduino y las baterías, y en la parte externa se sitúa la cámara de captura de video.

Para el aterrizaje se utilizó un paracaídas hecho de tela impermeable de color morado, que proporciona resistencia al viento y a la humedad. La elección de un color visible y llamativo es para poder identificar y recuperar fácilmente el SmartSAT 2.0. El paracaídas está sujeto al cohete de agua con el que se lanza el cansat y es controlado por un servo motor, el cual gira después de 10 segundos y permite la apertura de una puerta que libera el paracaídas.

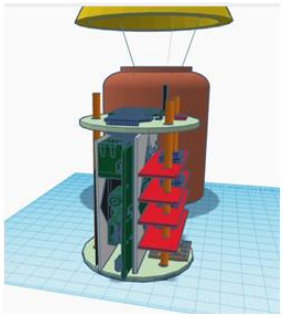


Fig. 8. Render 3D Cansat SmartSAT 2.0

Fig. 9. Servo motor SG90

4.4 Sistema de comunicación:

El sistema cuenta con dos módulos XBee Pro, que tienen un alcance de hasta 1.6 Km en línea de vista. Estos módulos operan a una frecuencia de 2.4 GHz y pueden transmitir con una velocidad de 250 kbps. Para el sistema de transmisión, se utilizará la frecuencia de 2.4 GHz que hace parte de las bandas para uso libre en Colombia, establecidas en la Resolución 711 de 2016 de la Agencia Nacional del Espectro (ANE). El módulo presente en el arduino uno está configurado como Router y se encarga de realizar la transmisión de los datos leídos por los sensores a la estación en tierra. Allí son recibidos por otro módulo XBee configurado como coordinador y conectado a un computador que muestra los datos en tiempo real. El diseño del SmartSAT 2.0 se realiza teniendo en cuenta la reglamentación de la Agencia Espacial Europea (ESA) El siguiente diagrama muestra el funcionamiento del sistema de comunicación del SmartSAT 2.0 con la estación en tierra:

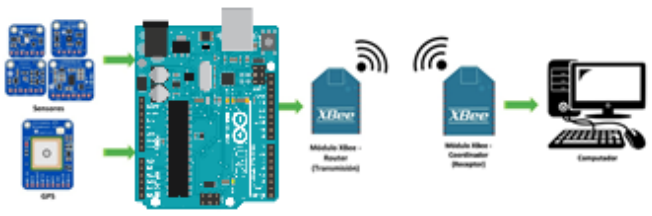


Fig. 10. Imagen funcionamiento del sistema de comunicación

5. costos smartsat 2.0

Los componentes seleccionados cumplen con la reglamentación de CanSat, pues no excede los 500 Euros y tiene un peso de 150 gr.

PRESUPUESTO SMARTSAT 2.0			
MATERIALES	CANTIDAD	COSTO UNITARIO	COSTO TOTAL
SISTEMA DE COMUNICACIONES			
BATERÍAS DE LITIO 3.7 V 800mAh	2	$17.000	$34.000
MÓDULO GPS6MV2	1	$65.600	$65.600
SENSOR BAROMÉTRICO BMP 280	1	$44.990	$44.990
SENSOR TEMPERATURA DTH11	1	$8.000	$8.000
SENSOR UV GYML 8511	1	$20.000	$20.000
SERVO MOTOR	1	$9.500	$9.500
PLACA ARDUINO	1	$18.000	$18.000
CABLES PARA ARDUINO	1	$4.000	$4.000
BATERIA DE 9V	1	$5.000	$5.000
ESTRUCTURA EN 3D	1	$30.000	$30.000
LATA DE GASEOSA	1	$2.500	$2.500
MINI CAMARA	1	$44.300	$44.300
MÓDULOS XBEE	2	$106.260	$212.520
SISTEMA DE PROPULSIÓN			
BOTELLAS DE GASEOSA	8	$2.500	$20.000
VÁLVULA	1	$8.000	$8.000
TUBOS PVC DE GAS	3	$10.000	$30.000
PEGANTE PARA TUBOS	1	$4.000	$4.000
LIMPIADOR DE TUBOS	1	$3.000	$3.000
TELA SOMBRILLA	1	$6.000	$6.000
MIPLE DE CARRO	2	$2.000	$4.000
RACOR DE COMPRESOR	2	$4.000	$8.000
COMPRESOR PEQUEÑO	1	$40.000	$40.000
AMARRADERAS	5	$200	$1.000
UNIÓN PARA MANGUERAS	1	$2.500	$2.500
UNIONES PARA TUBO	2	$1.000	$2.000
HOJA DE SEGUETA	1	$2.500	$2.500
PALOS DE BALSO	2	$600	$1.200
SELLADOR DE TANQUES	1	$16.000	$16.000
EMPAQUES	4	$700	$2.800
LÁMINA POLIESTIRENO	2	$13.100	$26.200
CINTA GRIS	1	$40.000	$40.000
VALOR TOTAL PROYECTO			$715.610

Tabla1. Costo del proyecto

6. configuraciones de software y pruebas de funcionamiento

6.1 Arduino nigthly

Es una plataforma open source para crear entornos, aplicaciones u objetos interactivos, existen diferentes tipos de placa dependiendo la capacidad o tipo de proyecto que se quiera llevar a cabo.

Para la programación de los componentes se utilizó comunicación serial, se realizó el código por medio de funciones dado que se necesitaban ejecutar varias tareas al mismo tiempo y se repitieran de forma constante, en el caso del servo motor el código solo se ejecuta una vez para que libere el paracaídas.

Pruebas y resultados:

1. Inicialmente se programaron todos los sensores sin funciones y se visualizaron los resultados ejecutándose directamente desde la interfaz de arduino obteniendo resultados satisfactorios.

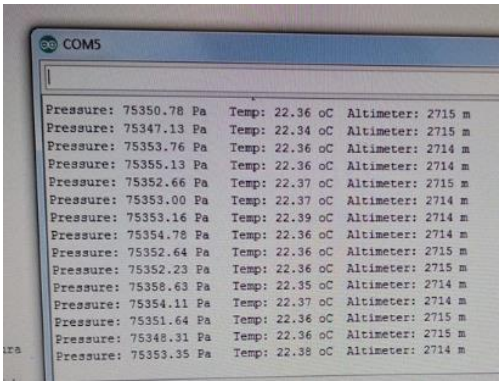


Fig. 11. Vista de datos sin funciones

6.1.1 Código DTH11

```
#include "DHT.h"
#define DHTPIN 7
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

int temperatura = 0;
int humedad = 0;

void setup() {
Serial.begin(9600);
dht.begin();
}
void loop() {

int humedad = dht.readHumidity();
int temperatura = dht.readTemperature();

Serial.print("Humedad: ");
Serial.print(humedad);
Serial.println("%");
Serial.print(" Temperatura: ");
Serial.print(temperatura);
Serial.println("°C");
delay (500);

}
```

Fig. 12. Código sensor DTH11

Para este sensor se incluye la librería DTH.h, se utilizó el pin 7 de conexión además de los pines de voltaje (5V) y GND, se define el tipo de sensor y las variables, se inicializa el puerto serial y se procede a tomar los datos de temperatura y humedad con un retardo de 500 ms.

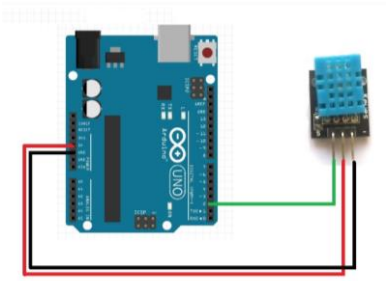


Fig. 13. Conexión DTH11 a arduino

6. 1.2 Código BMP280

```
#include <Adafruit_Sensor.h>
#include "Adafruit_BMP280.h"

Adafruit_BMP280 bmp;

float presion;
int altura;

void setup() {
  Serial.begin(9600);
  bmp.begin();
}

void loop(){
  presion = bmp.readPressure();
 altura= bmp.readAltitude ();
 Serial.print(F("Presión: "));
 Serial.print(presion);
 Serial.println(" Pa");
 Serial.print("Altura: ");
 Serial.print(altura);
 delay(500);
}
```

Fig. 14. Código sensor BMP280

Para este sensor se incluyen 2 librerías Adafruit_Sensor.h y Adafruit_BMP280.h, se utilizan los pines SCK y SDI conectados a los pines A5 y A4 respectivamente, además de los pines de voltaje (5V) y GND, se define el tipo de sensor y las variables, se inicializa el puerto serial y se procede a tomar los datos de Presión atmosférica y altura, con un retardo de 500 ms.

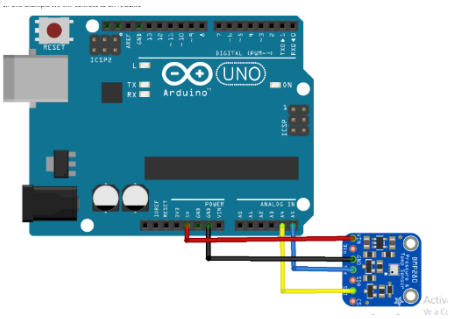


Fig. 15. Conexión sensor BMP280 a arduino

6.1.3 Código GYML 8511

```
int UVOUT = A0;
int REF_3V3 = A3;

void setup()
{
  Serial.begin(9600);
  pinMode(UVOUT, INPUT);
  pinMode(REF_3V3, INPUT);
  Serial.println("UV SmartSAT");
}

void loop()
{
  int uvLevel = averageAnalogRead(UVOUT);
  int refLevel = averageAnalogRead(REF_3V3);
  float outputVoltage = 3.3 / refLevel * uvLevel;
  float uvIntensity = mapfloat(outputVoltage, 0.99, 2.9, 0.0, 15.0);

  Serial.print(" UV Intensity (mW/cm^2): ");
  Serial.print(uvIntensity);
  Serial.println();

  delay(100);
}
```

Fig. 16. Código sensor GYMLl 8511 parte 1

```
int averageAnalogRead(int pinToRead)
{
  byte numberOfReadings = 8;
  unsigned int runningValue = 0;

  for(int x = 0 ; x < numberOfReadings ; x++)
    runningValue += analogRead(pinToRead);
  runningValue /= numberOfReadings;

  return(runningValue);
}

float mapfloat(float x, float in_min, float in_max, float out_min, float out_max)
{
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
```

Fig. 17. Código sensor GYMLl 8511 parte 2

Para el código de este sensor no fue necesario incluir ninguna librería, a diferencia de los sensores anteriores, éste trabaja con un voltaje de 3.3V y para la conexión se tuvo que realizar un puente entre el Pin EN y el pin de 3.3V, se declararon las variables, se inicializan, para este sensor fue necesario realizar un ciclo for que tomara el promedio de las lecturas y lo retornara

y también fue necesario usar la instrucción map para utilizar rangos de entrada y salida de voltaje necearios paracalcular la intensidad de radiación UV.

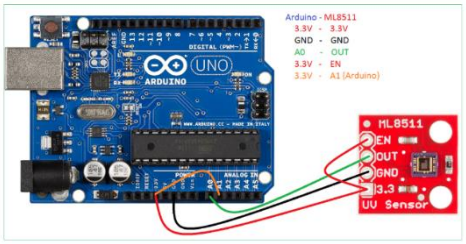


Fig. 18. Conexión sensor GYML 8511 a arduino

6.1.4 GY GPS6MV2:

```
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
#include <TinyGPS++.h>

SoftwareSerial gpsSerial(4,3);
TinyGPSPlus gps;

float lattitude,longitude;

void setup() {

gpsSerial.begin(9600);
Serial.begin(9600);

}

void loop(){

  int data = gpsSerial.read();
  lattitude = (gps.location.lat());
  longitude = (gps.location.lng());

 Serial.print ("lattitude: ");
 Serial.println (lattitude);
 Serial.print ("longitude: ");
 Serial.println (longitude);
 delay(500);
}
```

Fig. 19. Código sensor GY GPS6MV2

Para este sensor se incluye la librería TinyGPS plus, se declaran los pines a usar para este caso 3 y 4, los datos de latitud y longitud se declaran como tipo float, se inicializa el puerto serial y

se escriben los comandos para toma de datos de latitud y longitud, con un retardo de 500 ms, el sensor se alimenta a 5V y también un pin se conecta a tierra.

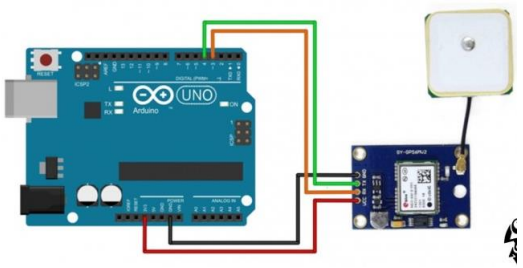


Fig. 20. Conexión sensor GY GPS6MV2

6.1.5 Servo SG90:

```
#include <Servo.h>
Servo serl;


void setup() {
serl.attach(9, 650, 2400);

}

void loop() {
serl.write(0);
delay(3000);
serl.write(120);
delay(9000);

}
```

Fig. 21. Código servo motor SG90

Para este motor se incluyó la librería servo.h, en el setup se declara el pin a usar en este caso el pin 9, y se indican el pulso mínimo y máximo del motor, en el write del loop se indican los grados que va a girar la hélice del motor en este caso de 0 a 120 grados. Para la conexión del motor con el arduino se realizó una conexión adicional a una batería de 9V ya que require mayor amperaje para trabajar, el pin 9 se conecta directo al arduino, el voltaje (5V) va a ala pila de 9V y la tierra se conecta a la pila de 9V y al pin GND del arduino.

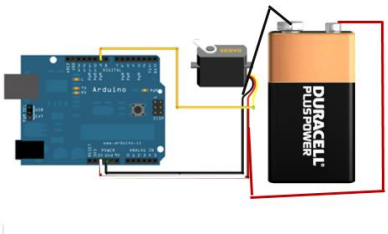


Fig. 22. Conexión servo motor SG90 a Arduino

6.2 MÓDULOS XBEE

```
#include <SoftwareSerial.h>

SoftwareSerial XBee(3,2);

void setup() {

Serial.begin(9600);
XBee.begin(9600);

}

void loop() {

if (XBee.available() > 0) {

int dato =XBee.read();

Serial.write(char(dato));

  }

}
```

Fig. 23. Código módulos Xbee

6.2.1 Para el módulo coordinador fue necesario incluir la librería SoftwareSerial, se indican los pines decdatos de la xbee a utilizar en este caso 2 y 3 los cuales se conectan a los pines Rx y Tx del arduino respectivamente, se inicializan los puertos y se hace una validación por medio de una sentencia if, si el módulo xbee recibe datos procederá a mosrarlos en pantalla, este módulo requiere alimentación de 3.3V.

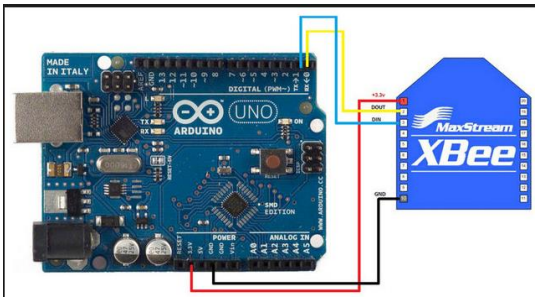


Fig. 24. Conexión módulos Xbee a arduino

6.2.2 Se hicieron pruebas incluyendo el código de los módulos xbee desde otra ventana, pero al compilar se evidenciaba un error que luego de investigar arduamente se evidenció que se presentaba debido a que en el administrador de dispositivos del puerto COM debía configurarse los drivers para que se reconociera el puerto COM del arduino.

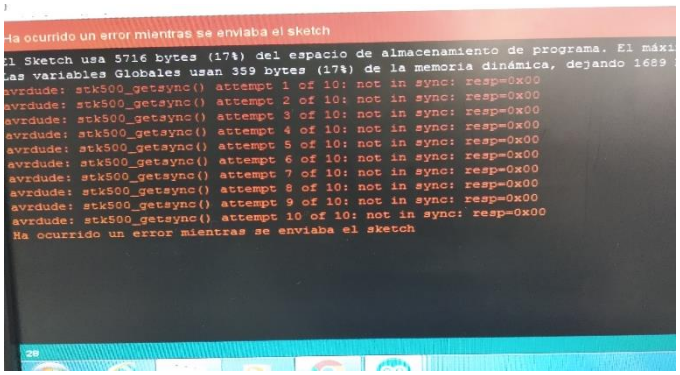


Fig. 25. Error de envìo de sketch para mòdulos X-bee

6.2.3. Se hicieron pruebas con el error corregido y se evidenció el envío de los datos.

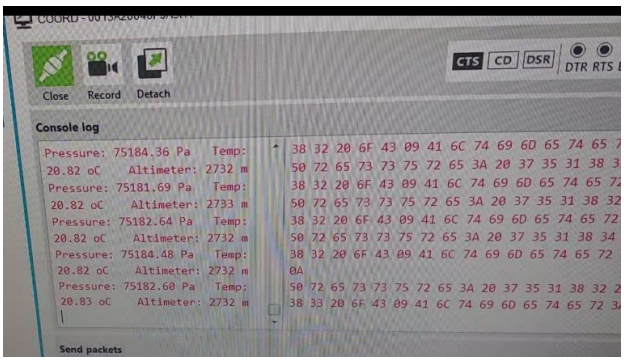


Fig. 26. Transmisiòn inalàmbrica desde XCTU

6.2.4. Al incluir el código de las xbee y el servo se hizo necesaria la creación de funciones, allí se pudo evidenciar que el servo generaba conflicto ya que el tiempo programado para que liberara el paracaídas era mayor que el tiempo que los sensores iban a enviar los datos y se tardaba mucho el envío de la información, así que se tuvo que modificar la función del servo y llamarla desde el setup, para permitir la apertura de la puerta del paracaídas se programó un tiempo de 10 segundos al inicio del Setup.

```
void loop() {

hiloUno();
delay(20);
hiloDos();
 delay(20);
hiloTres();
 delay(20);
}
```

Fig. 27. Llamado de funciones desde el void loop

```
void setup() {

delay(10000);
 bmp.begin();
 Serial.begin(9600);
 dht.begin();
 pinMode(UVOUT, INPUT);
 pinMode(REF_3V3, INPUT);
 gpsSerial.begin(9600);
 XBee.begin(9600);
 Serial.println("SmartSAT 2.0");
 servo();
}
```

Fig. 28. Llamado de función servo ( ) desde setup del código

6.2.5. Cuando se creó la base de datos se realizó una nueva función para guardar las lecturas en un archivo txt desde el cual se pretendía enviar la información a la base de datos, pero al no tener comunicación directa desde el arduino a internet no funcionó este código

```
#include <SPI.h>
#include <SD.h>

File myFile;


void setup() {
 while (!Serial) {
    ;

void loop(){
  myFile = SD.open("C:\Write\text.txt", FILE_WRITE);
  Serial.print("Writing to test.txt...");
   myFile.println(lattitude);
   close the file:
   myFile.close();

}
```

Fig. 29. Código inicial para base de datos

6.2.6. Después de observar que el código de la base de datos no funcionó se decidió realizar un programa que tomara los datos leídos directamente del XCTU y de allí se enviara un log a la base de datos pero como se explicará más adelante los datos deben ser leídos en números decimales para hacer más fácil la conversión de hexadecimal a ASCII, por ello se tuvo que modificar el código y quitar los print de los encabezados.

Postriormente se siguen leyendo los datos desde cada función pero únicamente se muestran datos decimales separados por comas.

```
Read fail0.00,-21206,0,0,24.43,4.63,-74.06
Read fail0.00,-21206,0,0,23.22,4.63,-74.06
Read fail0.00,-21206,0,0,22.89,4.63,-74.06
Read fail0.00,-21206,0,0,22.79,4.63,-74.06
Read fail0.00,-21206,0,0,22.42,4.63,-74.06
Read fail0.00,-21206,0,0,22.49,4.63,-74.06
Read fail0.00,-21206,0,0,22.46,4.63,-74.06
Read fail0.00,-21206,0,0,22.68,4.63,-74.06
Read fail0.00,-21206,0,0,22.26,4.63,-74.06
Read fail0.00,-21206,0,0,22.53,4.63,-74.06
Read fail0.00,-21206,0,0,22.62,4.63,-74.06
Read fail0.00,-21206,0,0,22.34,4.63,-74.06
Read fail0.00,-21206,0,0,22.45,4.63,-74.06
```

Fig. 30. Impresión de datos iniciales y posteriores sin encabezado

6.3 Fire base, node js y visual studio code

Firebase es una plataforma de desarrollo de aplicaciones web y aplicaciones móviles con una gran cantidad de servicios como lo son: Analítica (Firebase analytics), Desarrollo (Firebase cloud messaging, Fire auth, Realtime Database, Firebase Storage, Firebase Firestore).

La base de datos se creó con firebase y se programó para usarla como sistema contra incendios en el editor de código fuente visual sudio code con lenguaje phyton con ayuda del entorno Javascript basado en eventos Node js.

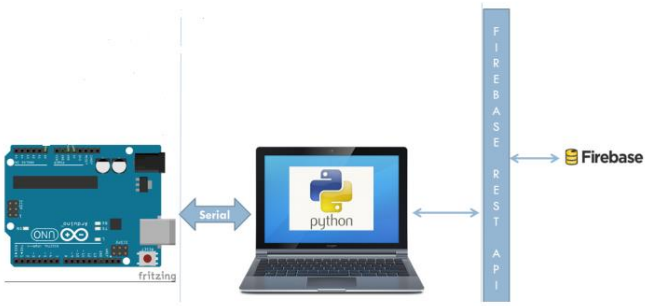


Fig. 31. Comunicación con base de datos contra incendio

6.3.1. Se instala el programa node Js y posteriormente desde la ventana de comandos se instala el express JS,es un framework para crear apps rápidas el cual nos permite manejar un modelo vista controlador, primero creó una carpeta en el PC en donde se guardará el proyecto de la base de datos contra incendio, en este caso la cramos en disco C con nombre projectfire, una vez iniciada la ventana de comandos abre desde allí la ubicación de la carpeta y se indicó el comado npm install express-generator -g para crear el repositorio global.

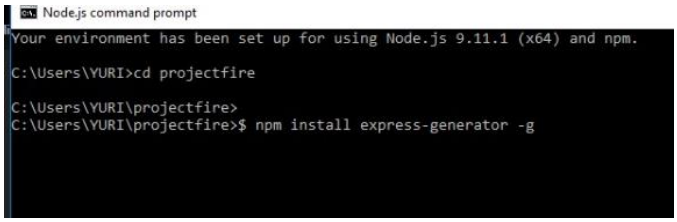


Fig. 32. Comando de repositorio global

6.3.2. Para generar el proyecto se digita el comando express --view=ejs firedetection dentro de la carpeta projectfire, "firedetection" hace referencia al nombre del proyecto.

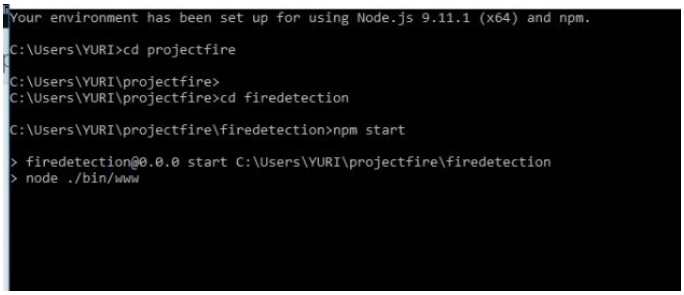


Fig. 33. Comandos de creación de rutas de carpetas

6.3.3. Una vez creado el proyecto se ingresa a la carpeta creada y se digita npm install y posteriormente npm start para iniciar el servidor.

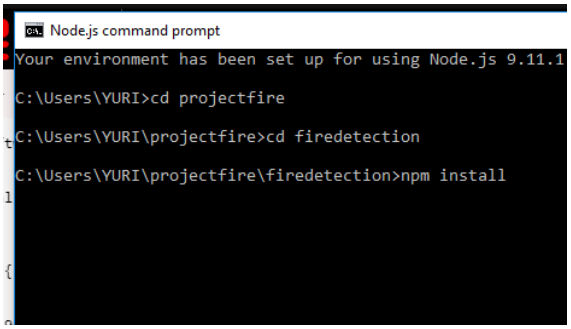


Fig. 34. Comando para instalar servicios de express js

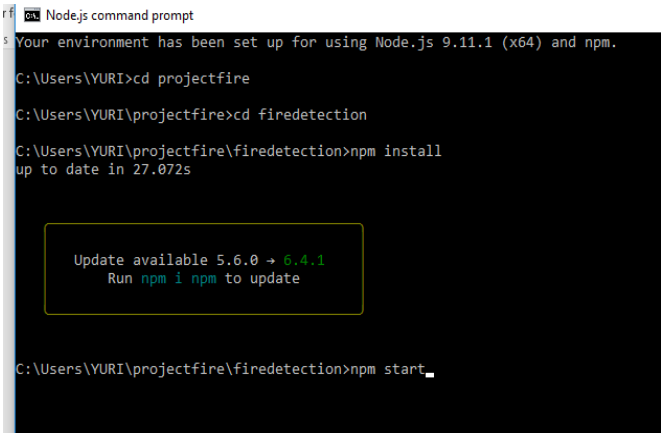


Fig. 35. Comando para iniciar el servidor

6.3.4. Para crear el código de la base de datos contra incendios se usó el editor de texto Visual Studio Code, en éste abrimos la carpeta del proyecto contra incendios creada en el disco C "firedetection", en la carpeta vistas se modifica inicialmente el código de index.ejs, allí se agrega el código para poder obtener el mapa de google, y se agrega la biblioteca multiplataforma bootstrap para hacer más amigable el entorno gráfico.

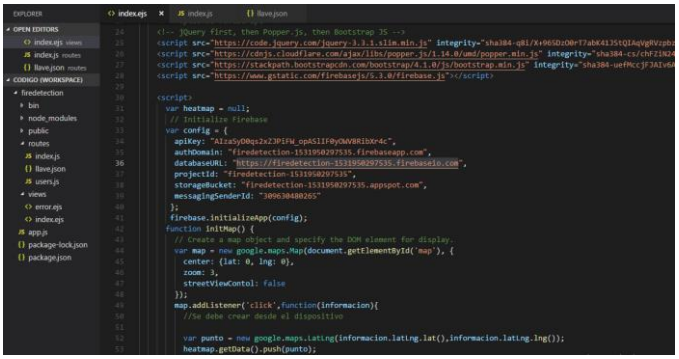


Fig. 36. Código base de datos contra incendio visual studio code

6.3.5. En este punto se hizo necesario solicitar un Api Key para que funcionara correctamente el mapa de google, dentro de la página se selecciona la opción obten una clave y se crea un nuevo proyecto allí, y se genera una clave, se copia y se pega en el código de Visual studio code donde dice YOUR_API_KEY, se añade la base de datos firebase al proyecto, una vez creada la cuenta en firebase se ingresa a la consola, allí se crea un proyecto, en este caso "", se selecciona la opción añadir fire base a aplicación web, aquí se copian las librerías para incluirlas en el index.ejs.

Fig. 37. Código de api key para añadir a index.ejs

6.3.6. Para hacer la conexión entre el servidor y firebase se deben habiitar los privilegios de administrador en el node js con el comando npm install firebase -admin --save

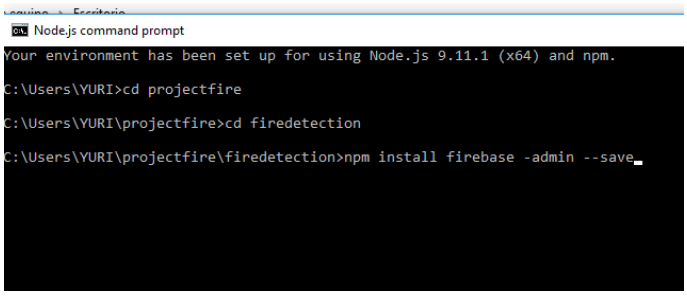


Fig. 38. Comando para habilitar privilegios de administrador

6.3.7. Se requiere descargar la configuración para el correcto funcionamiento de firease en el servidor, desde firebase en cuentas de servicio se genera una nueva clave privada, se descarga un archivo y se debe incluir en el editor de texto en index js.

Fig. 39. Clave privada para incluir en index js

6.3.8. Se crea el código post para recibir datos de latitud, longitud y estado (1 para incendio, 0 si no hay incendio)

```
router.get('/', function(req, res, next) {
  res.render('index', { title: 'Express' });
});

//POST
router.post('/informarIncendio',function(req,res){
  admin.database().ref('mapa/'+req.body.id).set({
    lat: req.body.lat,
    lng: req.body.lng,
    est: req.body.est


  });

  res.status(200).json({});
});
module.exports = router;
```

Fig. 40. Código para recepción de post

6.3.9. Para probar que el servicio esté funcionando se utiliza la aplicación postaman de google, allí se crea un post al localhost:3000 de la base de datos.

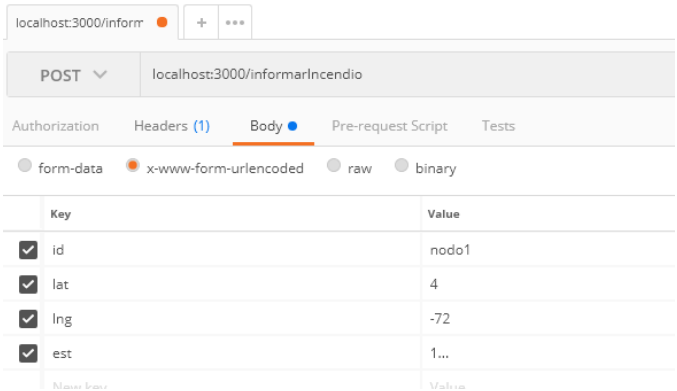


Fig. 41. Envío de post a base de datos contra incendio

De acuerdo con la norma NFPA (asociación nacional de protección contra el fuego) #13 en la tabla 2-2.4.1, una temperatura se considera alta a partir de los 107°C, por lo cual cuando firebase recibe temperaturas de 107°C en adelante deberá indicar con un circulo naranja que está ocurriendo un incendio, y para saber el punto exacto donde ocurre utiliza las coordenadas de latitud y longitud recibidas por el GPS.

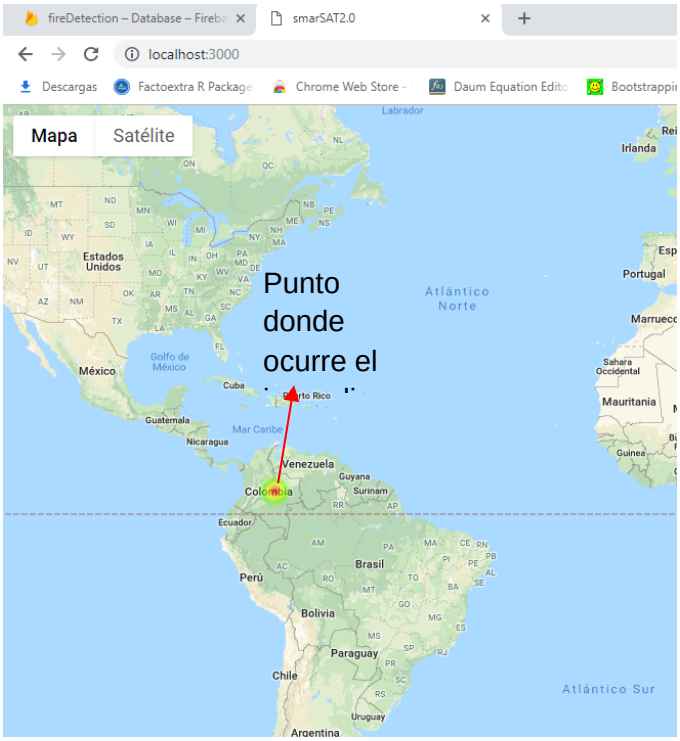


Fig. 42. Base de datos contra incendio

6.4 XCTU

Es el software usado por los módulos xbee para su comunicación, la interacción que hace entre las xbee y el usuario es por medio de una interfaz gráfica, en la cual se puede hacer la configuración de las xbee y la prueba de los módulos antes de ponerlos a funcionar en lo que deseamos, para este Proyecto utilizamos las antenas en modo AT lo cual quiere decir que los datos se envían de un módulo a otro sin tramas como sucede en el modo API.

Pruebas y resultados:

6.4.1. Al ejecutar el código de las xbee ya creado como función mostraba la información de los sensores, pero en forma de caracteres especiales.

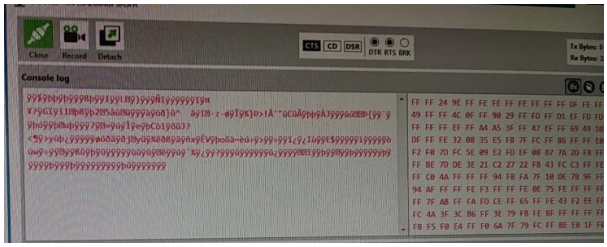


Fig. 43. Caracteeres especiales en XCTU al ejecutar el código con funciones

6.4.2. Debido a que se presentaron algunas inconsistencias con windows 10 se deicidió formatear el computador en el que se tenía toda la configuración e instalar el sistema operativo windows 7, luego de realizar este proceso al volver a transmitir la información al xctu se pudo observar sin caracteres especiales.

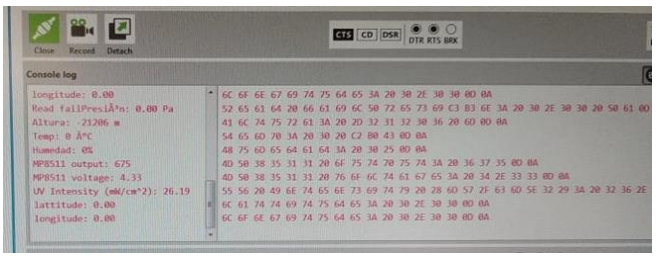


Fig. 44. Envío correcto de información a XCTU

6.4.3. Se realizaron capturas de logs para enviar la información a la base de datos, pero observamos que esta información se mostraba en hexadecimales.
Para guardar los logs desde la consola del módulo del coordinador se da click sobre el ícono record, se elige la ubicación donde se desea guardar el archivo txt y una vez se de clcick nuevamente sobre record quedará un archivo txt que se visualiza de la siguiente manera.

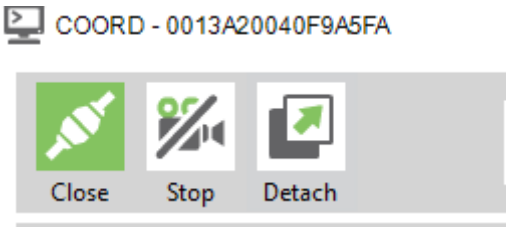


Fig. 45. Record del log

```
11-05-2018 20:24:04.342,-,AT,"COORD,0013A20040F9A5FA,UNKNOWN,4059,COM5

11-05-2018 20:24:04.921,159,RECV,52656164206661696C
11-05-2018 20:24:05.171,160,RECV,302E30302C2D32313230362C302C302C
11-05-2018 20:24:05.671,161,RECV,3031372E36352C
11-05-2018 20:24:06.234,162,RECV,342E36332C2D37342E30360D0A
11-05-2018 20:24:06.984,163,RECV,52656164206661696C
11-05-2018 20:24:07.234,164,RECV,302E30302C
11-05-2018 20:24:07.234,165,RECV,2D
11-05-2018 20:24:07.296,166,RECV,32313230362C302C302C
11-05-2018 20:24:07.734,167,RECV,30
11-05-2018 20:24:07.784,168,RECV,31362E39382C
11-05-2018 20:24:08.284,169,RECV,342E36332C2D37342E3036
11-05-2018 20:24:08.346,170,RECV,0D0A
11-05-2018 20:24:09.096,171,RECV,52656164206661696C
11-05-2018 20:24:09.346,172,RECV,302E30302C2D32313230362C302C302C
11-05-2018 20:24:09.846,173,RECV,3031392E34372C
11-05-2018 20:24:10.409,174,RECV,342E36332C2D37342E30360D0A
11-05-2018 20:24:11.221,175,RECV,52656164206661696C
11-05-2018 20:24:11.471,176,RECV,302E30302C2D32313230362C302C302C
11-05-2018 20:24:11.971,177,RECV,3032302E30312C
```

Fig. 46. Vista de log grabado

6.4.4. Luego de crear un programa en la base de datos para pasar datos de hexadecimal a decimal, se hizo necesario leer solamente datos decimales por lo cual se quitó el texto de las lecturas.

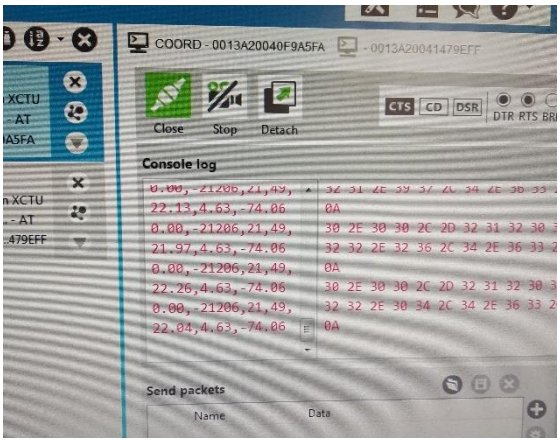


Fig. 47. Envío de datos en forma decimal separados por coma

6.5 Sistema de propulsión

El sistema de propulsión planteó como un cohete de agua realizado con materiales reciclables en este caso botellas plásticas, se creó un mecanismo con tubos para agua caliente en los cuales se bombea aire y en las botellas se deposita agua, esta combinación crea mucha presión en las botellas y al halar una válvula el cohete sale expulsado a una altura alrededor de los 50 mts, en la botella que contiene la punta del cohete se encuentra el paracaídas el cual está sujeto por una puerta plástica la cual está amarrada a un gancho del servo motor, cuando el servo gira el gancho la puerta se abre y se libera el paracaídas el cual está sujeto al centro de gravedad del cohete por medio de una pieza impresa en 3D para que no se enrede al girar.

pruebas y resultados:

6.5.1. Inicialmente se creó un sistema de propulsión con tubos PVC los cuales formaban una base y de allí salía verticalmente un tubo de 1 metro en el cual se incrustaba el cohete, pero el cohete tenía muchas fugas de agua y aire.

Fig. 48. Primer modelo de sistema de propulsión

6.5.2. El cohete de agua se creó con botellas de 1.75L de sprite pero por la forma que tienen la cinta no quedaba bien pegada y se plegaba, además se puso un racor metálico en medio de las botellas 2 y 3 pero había fuga de agua y al hacer la primera prueba se quedó un poco de agua allí.

Fig. 49. Primer modelo de sistema de cohete de agua

6.5.3. Para la puerta de expulsión del paracaídas se creó un gancho impreso en 3D al cual se sujeta un caucho, en la primera prueba al alimentar el servo con el cable de datos que estaba alimentando al arduino no logró mover la hélice así que se realizó la conexión de una pila de 9V para alimentar el servo de forma independiente y así funcionó correctamente.

Fig. 50. Piezas impresas en 3D para gancho de puerta y sujetador de cuerda del paracaídas

Fig. 51. Servo motor con conexión a pila de 9V

Fig. 52. Imágenes servo girando

Fig. 53. Imágenes expulsión de paracaídas

6.5.4. Se realizó un segundo sistema de propulsión más cómodo y con una válvula de carro para expulsar la botella, se realizó con tubos para agua caliente, al hacer la primera prueba con este sistema de propulsión se evidencio una fuga de aire en la válvula la cual se intentó solucionar con teflón.

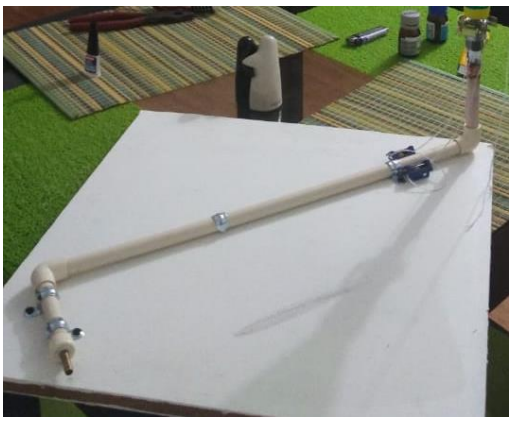


Fig. 54. Sistema de propulsión con tubos de agua caliente

6.5.5. Se creó un nuevo cohete de agua con botellas de contextura lisa y en lugar de un racor metálico se adicionó una tuerca, se sellaron todas las fugas de agua y aire con empaques de caucho.

Fig. 55. Segunda versión del cohete

6.5.6. Se realizó otra prueba de lanzamiento y aunque aparentemente no había fugas de aire ni agua el cohete no obtuvo suficiente presión para volar.

Fig. 56. Prueba de lanzamiento sistema de propulsión

6.5.7. Se adquirió un compresor de aire para poder inyectar mayor cantidad de aire al sistema, pero en el momento que ingresó mayor cantidad de aire se observaron nuevas fugas de aire, así que se le aplicó una mezcla química para pegar tanques.

Fig. 57. Compresor de aire a 12V

Fig. 58. Tubos con sellante para tanques

6.5.8. Se revisó que el sistema de propulsión no tuviera más filtraciones de aire y se hizo un primer lanzamiento con 70 psi de aire y aproximadamente 3L de agua pero el cohete quedó muy pesado y salió expulsado hacia un lado.

Fig. 59. Ingreso de aire al sistema de propulsión

Fig. 60. Expulsión del cohete hacia un lado

6.5.9. Se realizó un nuevo intento con una sola botella sin agregar agua y con 40 Psi de aire, se elevó cerca de 10 metros de altura pero chocó con un cable de la energía.

Fig. 61. Ingreso de aire a la botella

Fig. 62. Botella a 10 metros de altura aproximadamente chocando con cable de energía.

6.5.10. Se hizo un nuevo lanzamiento con el cohete, adicionando 1L de agua y 80 Psi de aire, el cohete sale expulsado por la presión pero se va hacia atrás y choca con un vehiculo, ocasionando fuertes daños en el cohete, los alerones se desprendieron y se dañaron.

Fig. 63. Expulsión del cohete hacia atrás

Fig. 64. Daños causados con el choque

6.5.11. Se reparó el cohete y se realizaron dos pruebas de lanzamietno con paracaídas con 0.5L de agua y 60 Psi de aire, en el primer lanzamiento se eleva aproximadame 2 metros y cae, en ell segundo lanzamiento el cohete gira sin elevarse y cae al suelo.

Fig. 65. Elevación alrededor de 2 mts de altura

Fig. 66. Cohete sin elevarse

6.5.12. Debido a que es muy difícil controlar la altura, y dirección del cohete de agua y para evitar daños irreparables en el satélite al usar este sistema de lanzamiento se decide lanzar el cohete desde una terraza aproximadamente de 15 metros de altura, con el paracaídas, en este lanzamiento se ralizó transmisión inalámbrica de las variables meteorológicas.

Fig. 67. Lanzamiento de cohete con paracaídas

Fig. 68. Descenso con paracaídas

6.6 Base de datos para mostrar las mediciones obtenidas

Conexión con servidor AWS:

Para la conexión a la base de datos se utilizó, primeramente, un servidor instanciado desde la capa gratuita de AWS con el fin de que, al tener la base de datos en un servidor virtual se pudiese cargar la información recibida por el dispositivo y así mismo varios usuarios previamente autorizados tuviesen acceso a los datos. Adicional, a partir de las bondades y el fácil manejo de los servicios de la consola de AWS logramos la conexión de una forma rápido a través del motor de base de datos utilizado en nuestro proyecto MySQL Workbench.

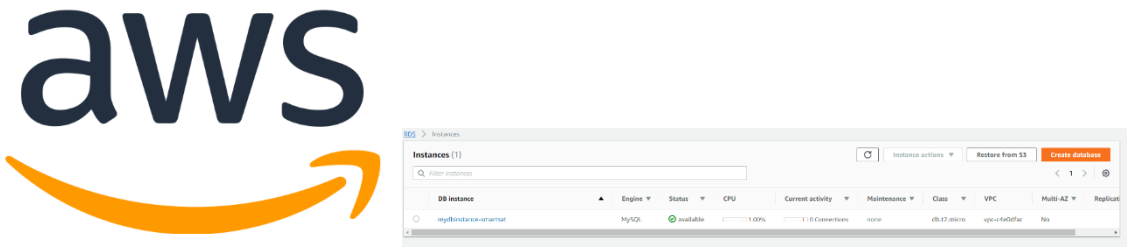


Fig. 69. Consola AWS

6.6.1.Para la creación de la instancia en la base de datos, y conectarla con nuestro servidor, se eligióel motor de la base de datos MySQL.

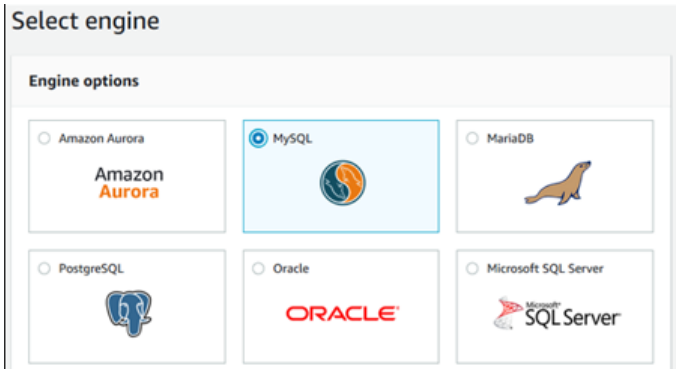


Fig. 70. Motor de base de datos MySQL

6.6.2. Ahora, es importante conocer a detalle los parametros que le daremos a la instancia para proceder con la creación en el servidor de la base de datos, desde el servidor se crea un usuario root que se encarga de tener el control total de la DB.
Adicional, se debe tener en cuenta el puerto donde procederemos a realizar la conexión con el fin de apuntar a una dirección y un puerto.

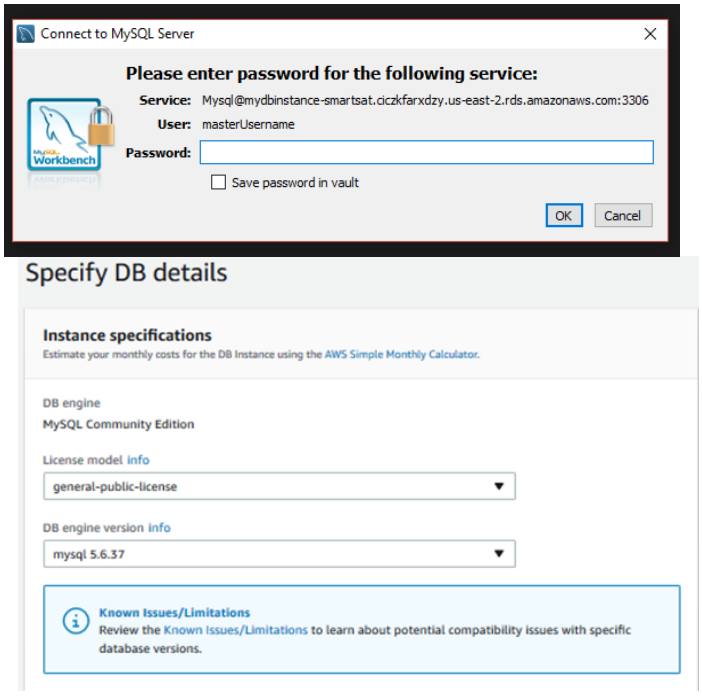


Fig. 71. Creación de parámetros de la base de datos

6.6.3. Una vez realizada la conexión al servidor, AWS dentro del service Database, RDS nos ofrece algunos componentes estadísticos y de funcionamiento para tener en cuenta dentro del funcionamiento de la base de datos en el servidor como:

a. El número de conexiones controladas en la base de datos junto con su utilización de CPU.

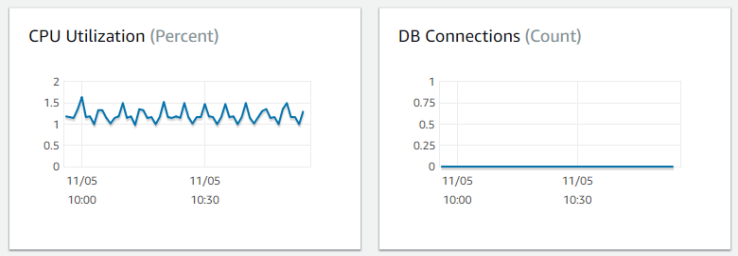


Fig. 72. Utilización y conexiones de la CPU

b. Número de operaciones de pruebas de lectura y escritura en las unidades denominadas como IOPS (Read – Write).

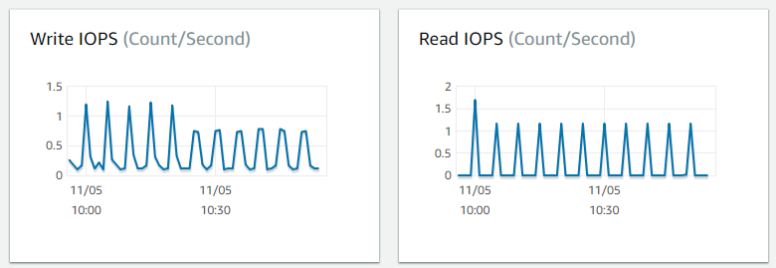


Fig. 73. Número de operaciones de lectura y escritura

Base de datos:

Como previamente lo habíamos mencionado nuestra base de datos se encuentra alojada dentro del motor de MySQL Workbench en la cual logramos una conexión

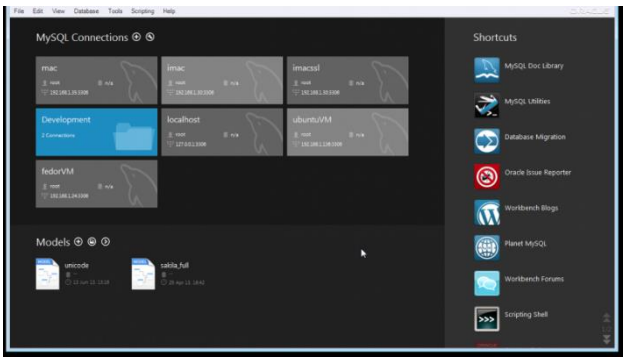


Fig. 74. MySQL Workbench

6.6.4. Se necesita un servidor independiente con distribución gratuita para abrir y direccionar el puerto que dará la conexión, por lo tanto, usamos XAMPP, una plataforma gratuita y de uso fácil para el usuario.

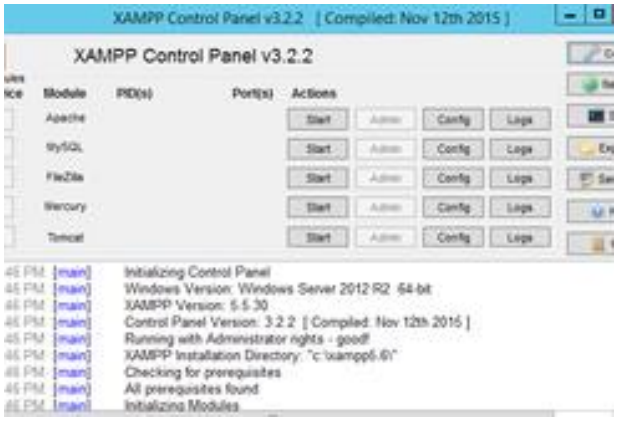


Fig. 75. Plataforma Xampp

6.6.5.Cuando se realizó la conexión se pudo ingresar al motor de DB apuntando le al Endpoint instanciado desde la base de datos en el servidor con un usuario root creado directamente desde el servidor.

6.6.6. La creación de tablas y tipos de variables que se utilizaron para recibir y cargar los datos fueron basados en el tipo y cantidad de datos que se esperaba recepcionar desde las antenas XBee, por consiguiente, se tuvo en cuenta el tipo de dato a recoger y el campo a llenar en la base de datos.

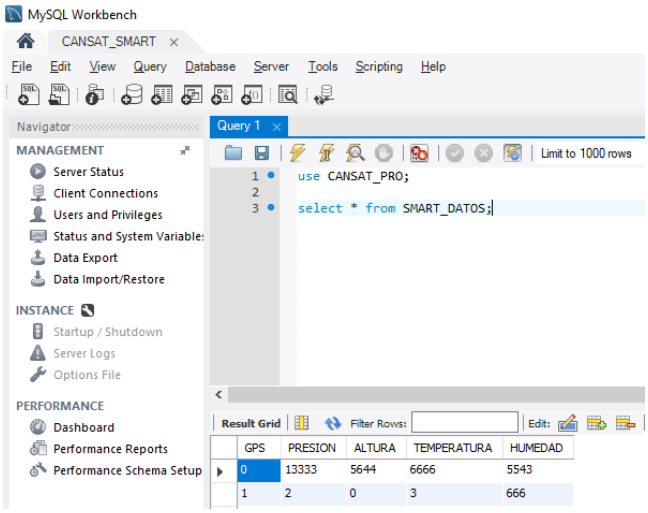
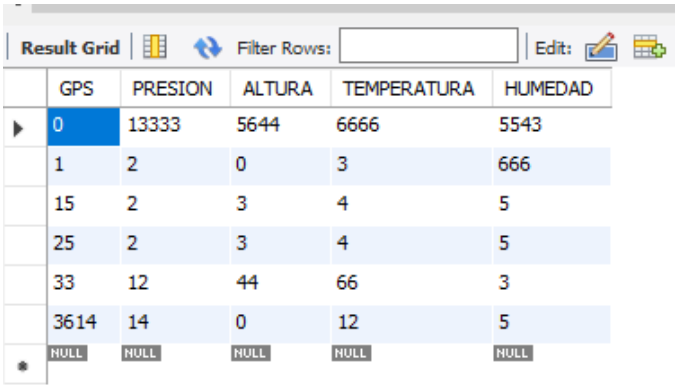


Fig. 76. Definición de tablas



GPS	PRESION	ALTURA	TEMPERATURA	HUMEDAD
0	13333	5644	6666	5543
1	2	0	3	666
15	2	3	4	5
25	2	3	4	5
33	12	44	66	3
3614	14	0	12	5
NULL	NULL	NULL	NULL	NULL

Fig. 77. Vista de tablas

Programa en java para conversión de datos:

Una vez se tenía la conexión y la base de datos, el proceso para cargar estos datos hizo necesario crear un programa que realizara la conexión a la base de datos y enviara los datos a los campos de la tabla, por consiguiente se creó un programa en java.

6.6.7. El programa se encuentra basado en lenguaje java en el ambiente de desarrollo de Eclipse, la necesidad se presenta desde que el log de datos enviado desde las antenas XBee se encuentra en formato hexadecimal, lo que conlleva a realizar el programa que lea el archivo y lo pase a formato ASCII para lograr su manipulación.

Primero, tuvimos que lograr que el programa leyera un archivo con formato texto e internamente pueda convertir en el formato ASCII para el pos envío a la base de datos y tener la información almacenada. Creamos una clase de tipo ReadFile para leer el archivo previamente mencionado.

.

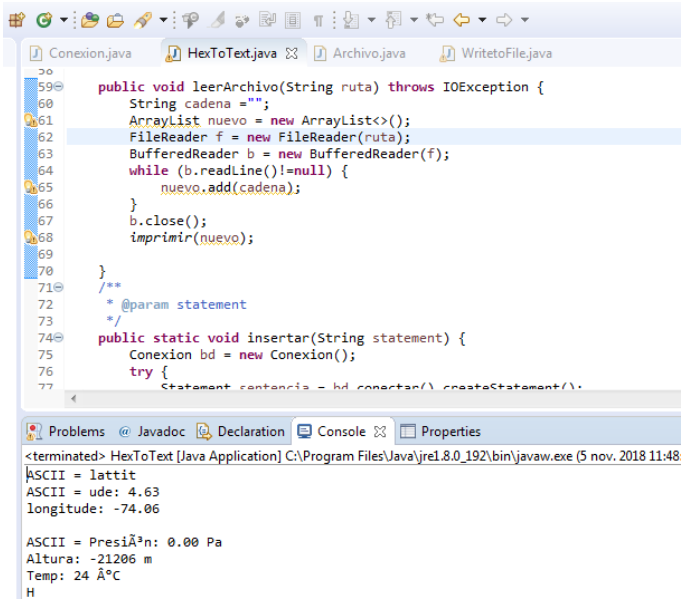


Fig. 78. Código para convertir a formato ASCII

6.6.8. Cuando se realiza su lectura, hace la conversión y envía los datos a la DB logrando una conexión desde java.

Fig. 79. Código para conexión con base de datos

6.6.9. En el proceso de la organización de los datos para insertarlos desde el programa se tornó con mayor dificultad por lo decidimos enviar los datos recogidos a un archivo plano y desde la DB realizar la importación de los datos recogidos, almacenarlos y dejar a la disponibilidad del usuario.

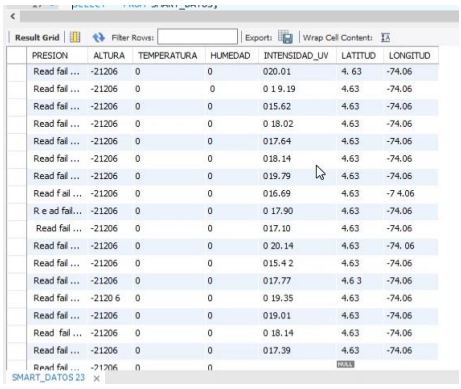


Fig. 80. Información en tablas de base de datos

6.7 Pruebas de peso y transmisión:

6.7.1. Se realizaron pruebas inalámbricas del satélite alejando el dispositivo a una distancia aproximada de 3 metros y no perdió conexión.

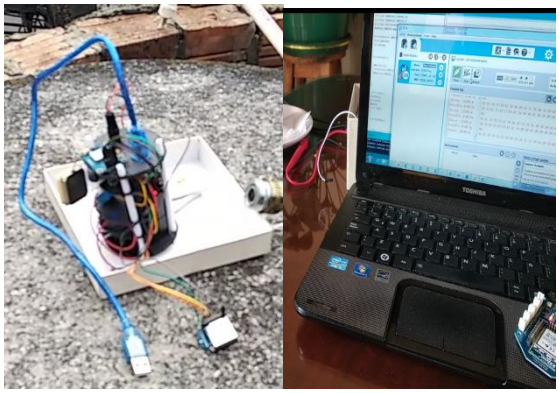


Fig. 81. Transmisión inalámbria a 3 mts aproximadamente

Fig. 82. Transmisión inalámbria a 15 mts aproximadamente

6.7.2. Se hizo una segunda prueba alejando el computador con el coordinador hasta el primer piso de la casa y de allí camimamos una cuadra, logrando una distancia aproximada de 15 metros sin linea de vista y transmitió perfecto al XCTU.

6.7.3. Se pesó el satélite y como se puede observar se obtuvo un peso total de 150gr, locual corresponde a la mitad del peso reglamentado para este tipo de satélite.

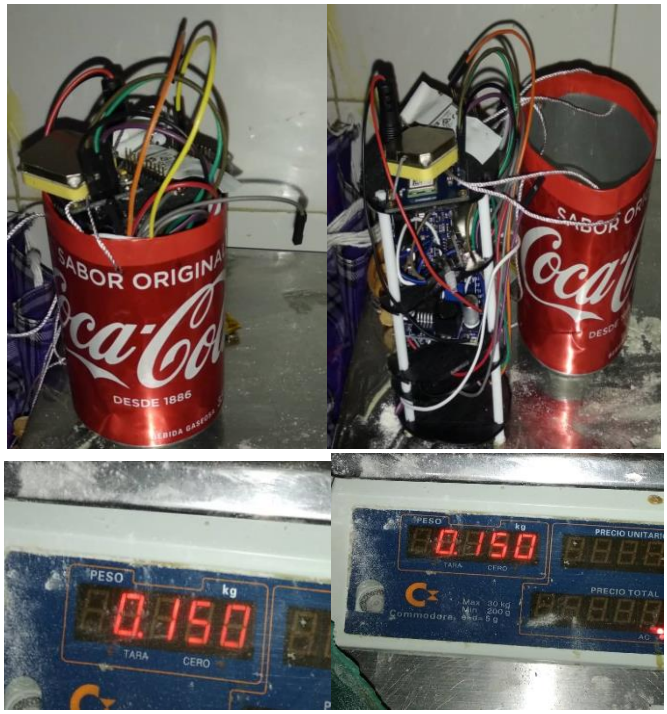


Fig. 83. Imagénes peso del SmartSAT 2.0

6.7.4. El SmartSAT 2.0 cuenta con una mini cámara de 2 mega pixeles, la cual puede capturar fotos y video con una batería que dura aproximadamente 1h con carga.

Fig. 84. Imagénes mini cámara y captura con mini cámara

7. Conclusiones

1. Algunas cosas resultaron más complejas de lo que se pensaba e incluso se hizo necesaria la creación de un programa en java para convertir hexadecimales a código ASCII debido a necesidades que solo surgieron cuando se estaba desarrollando el proyecto.

2. El sistema de propulsión no fue una buena idea como se había planteado, se tornó muy difícil poder controlar la altura y dirección a la cual volaría.

3. Se cumplió con la normatividad del Cansat a cabalidad de acuerdo con la reglamentación postulada en la Agencia Espacial Europea (ESA)

4. La misión meteorológica para la que fue diseñada el SmartSAT 2.0 funcionó corrrectamente permitiendo enviar los datos inalámbricamente y visualizarlos desde una base de datos ubicada en un servidor virtual.

8. Limitaciones

1. Inicialmente se había planteado realizar el proyecto con Raspberry Pi, pero los costos de esta placa incrementaban bastante el total del proyecto así que decidimos cambiarla por una placa de arduino Uno que para este proyecto nos servía sin inconvenientes y no se modificó nada en cuanto al funcionamiento que se planteó al inicio

2. Las baterías NCR 18650B se habían planteado en el ante proyecto, pero por costo y debido a que tocaba importarlas se decidió usar baterías de polimero litio que también funcionan a 3.6V.

3. Los sensores planteados eran de marca adafruit, por temas de costos y teniendo en cuenta que la universidad piloto de Colombia contaba con sensor de temperatura y barómetro, se decidió trabajar con estos sensores de marca sunfounder, pero conservando la cantidad y tipo de sensor descritos en el ante proyecto.

9. Referencias bibliográficas

K.N Manoj, K. Akhi, S. Kumar, M. Prathap (2016), *"Implementig smart home using firebase"*, IJREAS, Vol. No. 6, pp. 7. Recuperado de: http://euroasiapub.org/journals.php

H. Jean, B. Mauricio, P. (2014) Saúl, *"Diseño e implementación de un sistema Scada inalámbrico mediante la tecnología zigbee y arduino"*, prospect, vol. No 12, pp. 7. Disponible en: http://dx.doi.org/10.15665/rp.v12i2.290

H. Rafael (2009), *"guía de usuario de arduino"*, Universidad de Córdoba. Disponible en: http://www.uco.es/aulasoftwarelibre/wp-content/uploads/2010/05/Arduino_user_manual_es.pdf

B. Julien (2013), *"programming for arduino"*, bases de datos ebook collection.

Sriparasa, S. Srinivas (2013), *" javaScript and JSON essentials*", base de datos eBook collection, libro electrónico.

Kuppusamy, Prabhakaran (2014), *"Aws Essentials"*, base de datos eBook collection, libro electrónico.

W. Andreas Ragil, H. Khafiizh (2017*), "Implementation of Firebase Realtime Database to Track BRT Trans Semarang"*, Scientific Journal of Informatics, Vol 4, No 2. Disponible en: https://journal.unnes.ac.id/nju/index.php/sji/article/view/10829

Baeldung(2018), *"Convert Hex to ASCII in Java"*. Disponible en : https://www.baeldung.com/java-convert-hex-to-ascii

Datasheet batería polimero litio 3.7V: Disponible en: https://.www.batteryspace.com/prod-specs/NCR18650B.pdf

Regulador de voltaje LM2596. Disponible en: https://electronilab.co/tienda/modulo-lm2596-convertidor-de-voltaje-dc-dc-buck-1-25v-35v/

BMP pressure sensor module. Disponible en: https:// http://wiki.sunfounder.cc/index.php?title=BMP280_Pressure_Sensor_Module

DTH11 Humiture sensor. Disponible en: https://www.amazon.com/SunFounder-Humiture-Sensor-Arduino-Raspberry/dp/B014KOSFLC#feature-bullets-btf

UV Index Scale. Disponible en: https://www.epa.gov/sunsafety/uv-index-scale-1

Mófulo GPS GY GPS6MV2. Disponible en: https://amgkits.com/home/30-modulo-serial-gps-gy-neo6mv2.html

Polylactic Acid (PLA): The Environment-friendly Plastic. Disponible en: http://3dinsider.com/what-is-pla/

Resolución 711 de 2016 – Frecuencias de uso libre. Disponible en: http://cnabf.ane.gov.co/cnabf/modulos/pdfs/Resolucion_711_de_2016.pdf

Connecting XBee to Raspberry Pi. Disponible en: https://dzone.com/articles/connecting-xbee-raspberry-pi

What is a CanSat. Disponible en: https://www.esa.int/Education/CanSat/What_is_a_CanSat

Diseño de un sistema contra incendio con rociadores automáticos y cajetines de mangueras para un edificio de oficinas. Disponible en: https://www.dspace.espol.edu.ec/bitstream/123456789/16175/1/Dise%C3%B1o%20de%20un%20Sistema%20Contra%20Incendio%20con%20Rociadores%20Autom%C3%A1ticos%20y%20Cajetines.pdf

Qué es arduino nigthly. Disponible en: http://arduino.cl/que-es-arduino/

Firebase. Disponible en: https://firebase.google.com/?authuser=0

Creación de repositorio para expressJs. Disponible en: http://expressjs.com/es/starter/generator.html.

Biblioteca bootstrap. Disponible en : http://expressjs.com/es/starter/generator.html.

Mapas de google para javascript. Disponible en: https://developers.google.com/maps/documentation/javascript/tutorial?hl=es-419

XCTU para módulos XBee, disponible en: https://www.digi.com/resources/documentation/digidocs/90001458-13/default.htm

What is a cansat. Recuperado de: https://www.esa.int/Education/CanSat/What_is_a_CanSat